PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 12/08 — A1

(11) International Publication Number: WO 93/18459

(43) International Publication Date: 16 September 1993 (16.09.93)

(21) International Application Number: PCT/US93/01814

(22) International Filing Date: 3 March 1993 (03.03.93)

(30) Priority data:
07/847,300 6 March 1992 (06.03.92) US

(71) Applicant: RAMBUS INC. [US/US]; 2465 Latham Street, Mountain View, CA 94040 (US).

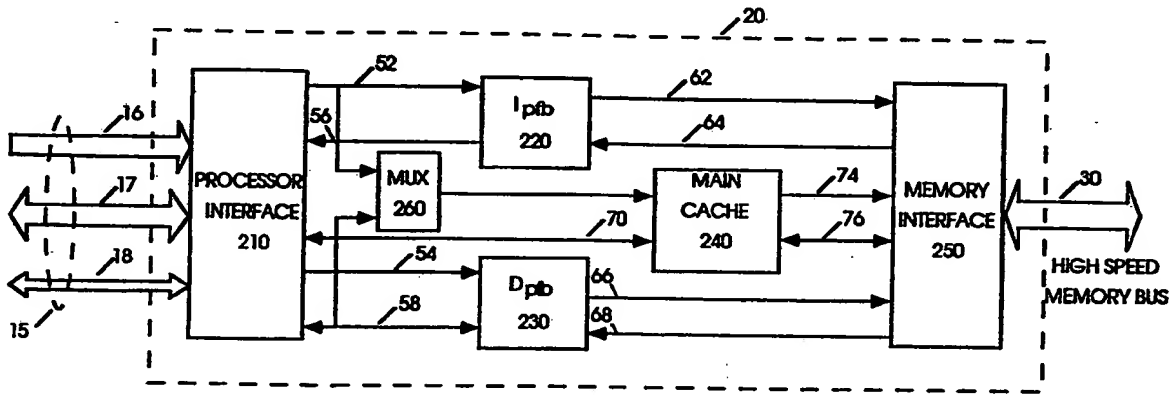
(72) Inventors: KRISHNAMOHAN, Karnamadakala ; 3168 Arcola Court, San Jose, CA (US). FARMWALD, Paul, Michael ; 190 Golden Oak Drive, Portola Valley, CA 94028 (US). WARE, Frederick, Abbott ; 13961 Fremont Pines, Los Altos, CA 94022 (US).

(74) Agents: VINCENT, Lester, J. et al.; Blakely, Sokoloff, Taylor & Zafman, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, CA 90025 (US).

(81) Designated States: AT, AU, BB, BG, BR, CA, CH, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, LK, LU, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, SN, TD, TG).

Published
With international search report.

(54) Title: PREFETCHING INTO A CACHE TO MINIMIZE MAIN MEMORY ACCESS TIME AND CACHE SIZE IN A COMPUTER SYSTEM



(57) Abstract

A cache memory subsystem with a relatively small main cache and a relatively high hit rate. Unpredictable data is stored in a main cache. Predictable data is stored in a set of instruction and data prefetch buffers. A cache subsystem with stride prediction hardware is also described.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FR	France	MR	Mauritania
AU	Australia	GA	Gabon	MW	Malawi
BB	Barbados	GB	United Kingdom	NL	Netherlands
BE	Belgium	GN	Guinea	NO	Norway
BF	Burkina Faso	GR	Greece	NZ	New Zealand
BG	Bulgaria	HU	Hungary	PL	Poland
BJ	Benin	IE	Ireland	PT	Portugal
BR	Brazil	IT	Italy	RO	Romania
CA	Canada	JP	Japan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SK	Slovak Republic
CI	Côte d'Ivoire	KZ	Kazakhstan	SN	Senegal
CM	Cameroon	LI	Liechtenstein	SU	Soviet Union
CS	Czechoslovakia	LK	Sri Lanka	TD	Chad
CZ	Czech Republic	LU	Luxembourg	TG	Togo
DE	Germany	MC	Monaco	UA	Ukraine
DK	Denmark	MG	Madagascar	US	United States of America
ES	Spain	ML	Mali	VN	Viet Nam
FI	Finland	MN	Mongolia		

PREFETCHING INTO A CACHE TO MINIMIZE MAIN MEMORY ACCESS TIME AND CACHE SIZE IN A COMPUTER SYSTEM

FIELD OF THE INVENTION:

The present invention pertains to the field of computer architecture. More particularly, this invention relates to cache memories systems for improving data access times.

BACKGROUND OF THE INVENTION:

Using dynamic random access memories ("DRAMs") for a high performance main memory for a computer system is often less expensive than using static random access memories ("SRAMs"). Nevertheless, DRAMs are typically much slower than SRAMs.

A common technique for lessening the impact of slow DRAM access time on main processor performance is to employ a cache memory. A cache memory is a limited size fast memory, usually made up of SRAMs, which stores blocks of data, known as lines, that reflect selected main memory locations. A cache memory is smaller than the main memory it reflects, which means the cache memory typically is not fully addressable and must store a tag field for each data line. The tag field identifies the main memory address corresponding a particular data line.

When the main processor issues a read request and an address corresponding to desired data stored in main memory, the cache memory is checked by comparing the received address to the tag fields of the cache memory. If the desired data is stored in the cache, then a "hit" occurs and the desired data is immediately available to the main processor. If the desired data is not stored in the cache, then a "miss" occurs, and the desired data must be fetched from slower main

memory. The typical goal in a cache memory design is to increase the hit rate because a low hit rate slows main processor performance.

One prior technique for increasing the hit rate in a cache memory subsystem is to use a prefetch buffer along with a main cache. A prefetch buffer is a memory that stores data prefetched from main memory. Data is speculatively prefetched into the prefetch buffer before a next read request based upon a prediction of the address for the next read request. When the main processor issues the next read request, the desired data may be available from the prefetch buffer if the prediction was accurate. In typical prior art systems, if the prediction was correct, the desired data is moved from the prefetch buffer to the main cache and is supplied to the main processor.

Nevertheless, prior art prefetch schemes that store prefetched data in the main cache often require relatively large main cache memories in order to maintain a high hit rate because the main cache typically becomes cluttered with predictable addresses, which are typically sequential. Unfortunately, larger cache memories increase the cost of the computer system and often preclude placement of effective caches on-chip with the main processor.

SUMMARY AND OBJECTS OF THE INVENTION

One object of the present invention is to minimize data access time in a computer system.

Another object of the present invention is to provide a relatively efficient cache bridge to a main memory.

Another object of the present invention is to minimize the size of the cache memory and to minimize memory access times.

Another object of the present invention is to provide a computer system with a relatively small cache memory with a relatively high hit rate that is comparable with the hit rates of larger cache memories.

A further object of the present invention is to provide an efficient cache memory subsystem suitable for placement on a microprocessor chip.

These and other objects of the invention are provided by a method and apparatus for reducing main memory access time in a computer system. In the cache memory subsystem of the present invention, an address corresponding to a data line stored in the main memory is received from a main processor, along with a read request. The address is received by an instruction prefetch buffer, a data prefetch buffer, and a main cache. If a hit occurs on one of the prefetch buffers, the data line is read from the prefetch buffer and transferred to the main processor. If a main cache hit occurs, then the desired data is read from the main cache and transferred to the main processor. If a main cache miss and prefetch miss occurs, then the desired data is fetched from main memory, stored in the main cache, and transferred to the main processor. In all cases (hit or miss), after the desired data has

been transferred to the main processor, a predicted address is generated. A next data line stored at the predicted address in the main memory is then fetched from the main memory and stored in the appropriate prefetched buffer. As a result, only data at unpredictable addresses are stored in the main cache. Data at predictable addresses do not clutter the main cache, but are instead stored in the instruction and data prefetch buffers.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings, and from the detailed description that follows below.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements, and in which:

Figure 1 is a block diagram of a computer system employing a separate cache subsystem;

Figure 2 shows a computer system with a cache subsystem on a processor chip;

Figure 3 is a functional block diagram illustrating the address and data paths for one cache memory subsystem;

Figure 4 is a logical flow diagram of the method employed by the cache memory subsystem;

Figure 5 is a block diagram of the hardware of a cache memory subsystem that employs a prefetch assist unit;

Figure 6 is a logical flow diagram of the method employed by the cache memory subsystem that has a prefetch assist unit;

Figure 7 illustrates stride prediction hardware of a cache subsystem.

DETAILED DESCRIPTION

Figure 1 is a block diagram of the architecture of computer system 5. Computer system 5 includes processor 10 for transferring address, data, and control information to and from cache subsystem 20 over bus 15. Cache subsystem 20 transfers information to and from main memory 40 over high speed bus 30. For the embodiment shown in Figure 1, cache subsystem 20 comprises circuitry external to processor 10. Cache subsystem 20 is functionally transparent to processor 10. In other words, processor 10 issues read requests and addresses over bus 15 as if processor 10 were directly connected to main memory 40.

As described in more detail below, cache subsystem 20 helps to maximize cache hits while minimizing cache space.

Figure 2 shows another embodiment of the present invention wherein cache subsystem 21 of computer system 7 resides within processor chip 11. Processor 11 is coupled to cache subsystem 21. Cache subsystem 21 is in turn coupled to main memory 41 via bus 31.

Figure 3 shows cache subsystem 20 of Figure 1. Cache subsystem 20 is comprised of processor interface 210, instruction prefetch buffer 220 (I_{pfb}), data prefetch buffer 230 (D_{pfb}), main cache 240, and memory interface 250. Processor interface 210 is coupled to received addresses from processor 10 over bus 16. Processor interface 210 transfers data to and from processor 10 over data bus 17, and transfers control information to and from processor 10 over control bus 18. Buses 16, 17, and 18 are part of bus 15 shown in Figure 1.

I_{pfb} 220 prefetches and buffers instructions for processor 10. I_{pfb} 220 is coupled to receive instruction addresses from processor interface

210 over address path 52, and transfer instructions to processor interface 210 over data path 56. I_{pfb} 220 is coupled to transfer predicted instruction addresses to memory interface 250 over address path 62 and receive next instruction lines over data path 64.

In a similar manner, D_{pfb} 230 prefetches and buffers data for processor 10. D_{pfb} 230 is coupled to receive addresses from processor interface 210 over address path 54 and transfer data to processor interface 210 over data path 58. D_{pfb} 230 transfers predicted data addresses to memory interface 250 over address path 66 and receives next data lines over data path 68.

Main cache 240 holds unpredictable instructions not prefetched by I_{pfb} 220 and unpredictable data not prefetched by D_{pfb} 230. Main cache 240 receives, through multiplexer 260, either instruction addresses 52 or data addresses 54 from processor interface 210. In case of a main cache 240 miss, main cache 240 transfers addresses 74 to memory interface 250 and transfers data 76 to and from memory interface 250.

Figure 4 is a flow diagram of a method employed by cache subsystem 20. At block 100, a read request is received by cache subsystem 20 from processor 10. Processor interface 210 receives the desired address over address bus 16 and receives a read request signal over control bus 18. The received address is routed (1) to I_{pfb} 220 over address path 52, and (2) to D_{pfb} 230 over address path 54, and (3) to main cache 240 through multiplexer 260. A signal on control bus 18 indicates whether the read request is for an instruction fetch or a data fetch.

If control signals 18 indicate an instruction fetch, the instruction prefetch buffer is checked for the desired instruction. This occurs when processor interface 210 transfers instruction address 52 to I_{pfb} 220. On the other hand, if control signals 18 indicate a data fetch, then the data prefetch buffer is checked when processor 210 transfers data address 54 to D_{pfb} 230. The received address - - either instruction address 52 or data address 54 - - is transferred to main cache 240 through multiplexor 260.

At decision block 110, a prefetch buffer or main cache hit is sensed. If a prefetch buffer hit occurs, then control transfers to block 120, wherein the desired line, either instruction or data, is read from the appropriate prefetch buffer, either I_{pfb} 220 or D_{pfb} 230. The prefetch buffer is also "popped" to make room for prefetched instructions or data. Alternatively, the prefetch buffer may not be popped if a larger prefetch buffer is employed. The desired line is then transferred to processor interface 210 over the appropriate data path, either data path 56 or 58. Thereafter, at block 130, processor interface 210 transfers the desired line to processor 10 over data bus 17.

At block 180, a next sequential line is prefetched from main memory 40 into the appropriate prefetch buffer. In the case of an instruction fetch indicated by control signals 18, a next sequential instruction address 62 is transferred from I_{pfb} 220 to memory interface 250. Memory interface 250 accesses main memory 40 over high speed bus 30 and transfers next sequential instruction line 64 to I_{pfb} 220. In the case of a data fetch indicated by control signals 18, a next sequential data address 66 is transferred to memory interface 250 and next sequential data line 68 is transferred from memory interface 250 to

D_{pfb} 230 after being fetched from main memory 40. Control then proceeds to block 190, which ends the read request sequence.

If a main cache hit occurs at decision block 110, then control is transferred to block 150, wherein the desired line is read from main cache 240 and supplied to processor 10. The desired line is transferred from main cache 240 to processor interface 210 over data path 70. Control then proceeds to block 180, wherein a next sequential line is prefetched as discussed above.

If a main cache or prefetch buffer hit does not occur at decision block 110, then control transfers to block 160, wherein the "missed" line is fetched into main cache 240. Address 74, which is either the received instruction address 52 or data address 54, is transferred to memory interface 250. After accessing main memory 40, memory interface 250 transfers the desired line 76 to main cache 240. The desired line is stored in main cache 240 and transferred to processor interface 210 over data path 70. Processor interface 210 transfers the desired line to processor 10 over data bus 17. Control then proceeds to block 180, wherein a next sequential line is then prefetched as discussed above.

Figure 5 illustrates cache system 22, which is another embodiment of the present invention. Cache subsystem 22 employs a prefetch assist cache ("PAC"). Cache subsystem 22 functions as a cache bridge between processor 10 and a high performance memory system 40. An example of a high performance main memory system 40 is set forth in PCT international patent application number PCT/US91/02590 filed April 16, 1991, published October 31, 1991, and entitled Integrated Circuit I/O Using a High Performance Bus Interface.

Cache subsystem 22 is coupled to processor 10 via bus 15. Cache subsystem 22 is coupled to main memory 40 via bus 30. Cache subsystem 22 is located between processor 10 and main memory 40. Cache subsystem 22 is "transparent" to processor 10. Cache subsystem 22 includes instruction prefetch unit 510, data prefetch unit 530, prefetch assist unit 520, main cache unit 540, and control unit 550. Address latch 502 receives addresses 16 from processor 10. Address latch 502 transmits address signals 516 to instruction prefetch unit 510, data prefetch unit 530, prefetch assist unit 520, and main cache unit 540. Address signals 516 are also received by increment register 570 and multiplexer 560. Data is transferred between processor 10 and instruction prefetch unit 510, data prefetch unit 530, and main cache unit 540 over bus 17.

Instruction prefetch unit 510 is comprised of a control section ("CFB CTL") 631 and a data storage section 632 comprised of (1) data 633 and (2) tags and compare ("TAGS & CMP") section 634. Control section 631 communicates with control unit 550 over signal lines 552. Data storage section 632 is organized as a set of four fully associative buffers of size 16 bytes each. On an instruction prefetch hit, the desired line is supplied to processor 10, and the entry is "popped" to make room for a new prefetched line. Entries are replaced on a least recently used basis.

Similarly, data prefetch unit 530 is comprised of a control section ("DFB CTL") 641 and a data storage section 642 comprised of (1) data 643 and (2) tags and compare ("TAGS & CMP") section 644. Control section 641 communicates with control unit 550 over signal lines 553. Data storage section 643 is organized as a set of four fully associative buffers of size 16 bytes each. On a data prefetch hit, the desired line is supplied

to processor 10 and the entry is "popped" to make room for a new prefetched line on a least recently used basis.

Main cache unit 540 includes a control section ("CACHE CTL") 651 that communicates with control unit 550 over signal lines 554. Main cache unit 540 also includes data storage section 652 comprised of (1) data 653 and (2) tags and compare ("TAGS & CMP") section 654. Data storage section 652 is organized as an 8 kilobyte 4-way set associative unified cache with least recently used replacement.

Prefetch assist unit 520 is comprised of (1) a control section ("PAC CTL") 661, (2) a data section 652 comprised of previous tags ("PREV TAGS") 653 and next tags ("NEXT TAGS") 654, (3) a last code address register ("CAR") 655, and (4) a last data address register ("DAR") 656. Data section 652 is organized as a 256 entry 4-way set associative cache with least recently used replacement.

CAR 655 and DAR 656 are used in conjunction with desired address 516 to create a new PAC entry in the data section of prefetch assist unit 520. A PAC entry in the data section is comprised of a predicted address ("NEXT TAG") and an associated tag field ("PREV TAGS") defined by the last instruction or data address. The CAR and DAR entries are created by storing desired address 516 in CAR 655 or DAR 656, depending upon whether control signals 18 indicate an instruction read or a data read sequence.

Cache subsystem 22 employs a relatively small main cache to achieve a relatively high hit rate and avoids having a much larger main cache. For example, when employed as a cache bridge to high performance main memory system 40 capable of transferring 500

Megabytes/second, cache subsystem 22 uses only an 8 kilobyte main cache.

Figure 6 is a flow diagram illustrating the method employed by cache subsystem 22. At block 300, a read request is received by cache subsystem 22 from processor 10. This occurs when address register 502 receives the desired address over address bus 16 and processor control unit 504 receives a read request over control bus 18. The desired address 516 is received by instruction prefetch unit 510, data prefetch unit 530, main cache unit 540, prefetch assist unit 520, increment register 570, and multiplexer 560.

At decision block 310, if control unit 550 senses an instruction prefetch buffer hit via bus 552 or a data prefetch buffer hit via bus 553, then control transfers to block 320, wherein the desired line - - either instructions or data - - is "popped" from the appropriate prefetch buffer and transferred to processor 10 over data bus 17.

At decision block 360, if prefetch assist hit on bus 551 is sensed by control unit 550, then control is transferred to block 370, wherein control unit 550 issues mux control 555, which causes multiplexor 560 to couple predicted address 521 to fetch address register 580. Control unit 550 also issues control signals 556 to signal a fetch cycle to memory control unit 586. Fetch address register 580 transmits fetch address 19, and memory control 586 transmits control signals 22 over high performance bus 30, which initiates a main memory 40 fetch cycle. The line corresponding to predicted address 521 is fetched from main memory 40. Line 20 is received by receive data register 582 and transferred to either instruction prefetch unit 510 or data prefetch unit 530 over data bus 17, under control of fill prefetch signal 552 or 553

issued by control unit 550, depending upon whether control signals 18 indicate an instruction or a data read sequence. Control then proceeds to block 410, which ends the read request sequence.

If prefetch assist hit on bus 551 is not sensed by control unit 550 at block 360, then control proceeds to block 380, wherein increment register 570 generates next sequential line address 571. Control unit 550 then issues mux control 555 to couple next sequential line address 571 to fetch address register 580. Control unit 550 also issues control signals 556 to signal a fetch cycle to memory control unit 586. Fetch address register 580 transmits fetch address 619, and memory control 586 transmits control signals 622 over high performance bus 30, which initiates a main memory 40 fetch cycle. The next sequential line is then fetched from main memory 40. Next sequential line 20 is received by receive data register 582. Next sequential line 20 is then transferred to either instruction prefetch unit 510 or data prefetch unit 530 over data bus 17, depending upon whether control signals 18 indicate an instruction or a data read sequence. This is done under the control of fill prefetch signal on bus 552 or bus 553 issued by control unit 550. Control then proceeds to block 410, which ends the read request sequence.

If a main cache hit indicated on bus 554 is received at decision block 310, then control transfers to block 340, wherein the desired line is read from main cache unit 540 and supplied to processor 10 over data bus 17. Control then proceeds to block 360, wherein prefetching is performed as discussed above.

If a prefetch buffer hit or main cache hit is not received at decision block 310, then control transfers to block 350, wherein the

"missed" line is fetched into main cache unit 540. This occurs when control unit 550 issues control signals 555, which cause multiplexer 560 to couple desired address 516 to fetch address register 580. Control unit 550 also issues control signals 556 to signal a fetch cycle to memory control unit 586. Fetch address register 580 transmits fetch address 619 and memory control 586 transmits control signals 622 over high performance bus 30, which initiates a main memory 40 fetch cycle. The missed line is returned on bus 620, stored in receive data register 582, and transferred to main cache unit 540 and processor 10 over data bus 17. Control unit 550 then issues a fill main cache signal on bus 554, which causes main cache 540 to create a new entry.

At block 390, a next sequential line is prefetched from main memory 40 into the appropriate prefetch buffer. As before, increment register 570 generates next sequential line address 571. Control unit 550 then issues mux control 555 to couple next sequential line address 571 to fetch address register 580. Fetch address register 580 transmits fetch address 619 and memory control 586 transmits control signals 622 over high performance bus 30, which initiates a main memory 40 fetch cycle. The next sequential line is fetched from main memory 40. Next sequential line 20 is received by receive data register 582. Next sequential line 20 is transferred to either instruction prefetch unit 510 or data prefetch unit 530 over data bus 17 under control of a fill prefetch signal on bus 552 or bus 553 issued by control unit 550.

At block 400, a new PAC entry is created when control unit 550 issues a create PAC signal on bus 551. The CAR and DAR of prefetch assist unit 520 are used in conjunction with desired address 516 to create a new PAC entry in the data section of prefetch assist unit 520. A

PAC entry is created in the data section by storing desired address 516 as the predicted address (NEXT TAG), along with an associated tag field (PREV TAGS) defined by the last instruction address CAR or data address DAR, depending upon whether control signals 18 indicate an instruction or a data fetch sequence. Control then proceeds to 410, which ends the read request sequence.

Figure 7 illustrates an example of stride prediction hardware provided by another embodiment of cache subsystem 21 of Figure 2. A stride predictor 703 stores the last few program counter ("PC") and virtual address ("VA") pairs received from processor 11 during load instructions, and checks for re-occurrences of PC values corresponding to a load instruction. If a PC value re-occurrence is detected by the stride predictor 703, then the corresponding VA value ("VA_{match}") is used to generate a predicted VA 795 for prefetching into a data prefetch buffer, such as D_{pfb} 220.

For one embodiment, stride predictor 703 resides on processor chip 11. Registers 701 and 702 are embedded within processor 11 and contain the PC and VA values, respectively. For this example, the last 5 PC and VA pairs are stored.

Stride predictor 703 receives VA 781 and PC 783 from processor 11, along with signal 780, which indicates a load instruction, i.e. a load from memory or a memory read. In this example, the stride predictor 703 is comprised of stages 735 through 739. For stage 735, register 710 stores VA 781 and register 720 stores PC 783, both under control of load signal 780. For stage 736, register 711 stores VA 790 received from register 710 and register 721 stores PC 792 received from register 720, both under control of load signal 780.

Stages 736 through 739 function similarly to stage 735. Each time load signal 780 indicates a load instruction, the VA 781 and PC 783 values are propagated and stored in stages 735 through 739 in a "first in first out" manner.

Comparators 750 through 754 compare the newest PC 783 to the PC output of each stage 735 through 739. For example, in stage 735 comparator 750 compares PC output 792 of register 720 with newest PC 783. If comparator 750 detects a match, then enable signal 791 causes driver 740 to couple the VA 790 from register 710 to VA_{match} 782. In a similar manner, comparator 751 tests the output of register 721 for a match with newest PC 783. If a match is detected by comparator 751, then the output of register 711 is coupled to VA_{match} 782 by driver 741.

Each stage 735 through 739 tests the newest PC 783 to their stored PC value. If any of the outputs of registers 720 through 724 match the newest PC 783, then the corresponding VA stored in registers 710 through 714 is coupled to VA_{match} 782.

Math logic 730 receives VA_{match} 782 and newest VA 781, and generates predicted VA 795 in accordance with the following equation:

Predicted VA = newest VA + (newest VA-VA match) = 2 x newest VA - VA_{match}.

For example, math logic 730 may in one embodiment comprise a simple adder with one input comprising a logically left shifted newest VA 781 (2 x newest VA), and the other input comprising the two's complement of VA_{match} 782. VA 795 can be used for prefetching into a data prefetch buffer, such as D_{pfb} 220.

In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A method for minimizing memory access time in a computer system, comprising the steps of:

(a) receiving an address corresponding to a data line stored in the main memory and receiving a read request from a main processor;

(b) transferring the address to a prefetch buffer, and a main cache;

(c) sensing a prefetch buffer hit signal from the prefetch buffer, and a main cache hit signal from the main cache, the prefetch buffer hit signal indicating whether the data line corresponding to the address is stored in the prefetch buffer, and the main cache hit signal indicating whether the data line corresponding to the address is stored in the main cache;

(d) if the prefetch buffer hit signal indicates the data line is stored in the prefetch buffer, reading the data line from the prefetch buffer, and transmitting the data line to the main processor, then proceeding to step (h);

(e) if the main cache hit signal indicates the data line is stored in the main cache, reading the data line from the main cache, and transmitting the data line to the main processor, then proceeding to step (h);

(f) transmitting the address to the main memory;

(g) receiving the data line from the main memory, and transmitting the data line to the main processor;

(h) generating a predicted address, fetching a next data line corresponding to the predicted address from the main memory, and storing the next data line in the prefetch buffer.

2. The method of claim 1, wherein step (g) further comprises the step of storing the data line in the main cache.

3. The method of claim 1, wherein the step of generating the predicted address in step (h) comprises incrementing the address.

4. A method for minimizing memory access time in a computer system, comprising the steps of:

(a) receiving a read request from a main processor and an address corresponding to a data line stored in the main memory;

(b) transferring the address to a prefetch buffer, a main cache, and a prefetch assist cache;

(c) sensing a prefetch buffer hit signal from the prefetch buffer, and a main cache hit signal from the main cache, the prefetch buffer hit signal indicating whether the data line corresponding to the address is stored in the prefetch buffer, and the main cache hit signal indicating whether the data line corresponding to the address is stored in the main cache;

(d) if the prefetch buffer hit signal indicates the data line is stored in the prefetch buffer, reading the data line from the prefetch buffer, and transmitting the data line to the main processor, then proceeding to step (f);

(e) if the main cache hit signal indicates the data line is not stored in the main cache, then fetching the data line from the main memory, transmitting the data line to the main processor, and updating the prefetch assist cache;

(f) sensing a prefetch assist cache hit signal from the prefetch assist cache, the prefetch assist cache hit signal indicating whether a predicted address corresponding to the address is stored in the prefetch assist cache;

(g) if the prefetch assist cache hit signal indicates the predicted address corresponding to the address is not stored in the prefetch assist cache, fetching a next sequential data line from the main memory and storing the next sequential data line in the prefetch buffer;

(h) if the prefetch assist cache hit signal indicates the predicted address corresponding to the address is stored in the prefetch buffer, fetching a next data line corresponding to the predicted address from the main memory and storing the next data line in the prefetch buffer.

5. The method of claim 4, wherein fetching the data line from the main memory and updating the prefetch assist cache in step (e) comprises the steps of:

(a) transmitting the address to the main memory;

(b) receiving the data line from the main memory, storing the data line in the main cache, and transmitting the data line to the main processor;

(c) storing the address in the prefetch assist cache such that the address is selected by a last address received from the main processor.

6. The method of claim 4, further comprising the steps of:

(i) receiving a program counter address, a virtual address, and a load instruction control signal from the main processor;

(j) storing the program counter address and the virtual address if the load instruction control signal is received from the main processor,

such that a last "n" program counter addresses and virtual addresses are stored;

(k) comparing the program counter address to each of the last "n" program counter addresses;

(l) if the program counter address equals one of the last "n" program counter addresses, then generating a predicted virtual address by multiplying a corresponding stored virtual address by 2, and adding the virtual address;

(m) if the program counter address equals one of the last "n" program counter addresses, fetching the next data line corresponding to the predicted virtual address from the main memory and storing the next data line in the prefetch buffer.

7. An apparatus for minimizing memory access time in a computer system, comprising:

means for receiving a read request from a main processor and an address corresponding to a data line stored in the main memory;

means for transferring the address to a prefetch buffer and a main cache;

means for sensing a prefetch buffer hit signal from the prefetch buffer, the prefetch buffer hit signal indicating whether the data line corresponding to the address is stored in the prefetch buffer;

means for reading the data line from the prefetch buffer and transmitting the data line to the main processor if the prefetch buffer hit signal indicates the data line is stored in the prefetch buffer;

means for sensing a main cache hit signal from the main cache, the main cache hit signal indicating whether the data line corresponding to the address is stored in the main cache;

means for fetching the data line from the main memory if the main cache hit signal indicates the data line is not stored in the main cache;

means for reading the data line from the main cache and transmitting the data line to the main processor;

means for fetching a next sequential data line from the main memory and storing the next sequential data line in the prefetch buffer.

means for generating a predicted address, fetching a next data line corresponding to the predicted address from the main memory, and storing the next data line in the prefetch buffer.

8. The apparatus of claim 7, further comprising:

means for transferring the address to a prefetch assist cache;

means for sensing a prefetch assist cache hit signal from the prefetch assist cache, the prefetch assist cache hit signal indicating whether a predicted address corresponding to the address is stored in the prefetch assist cache;

means for fetching the next sequential data line from the main memory and storing the next sequential data line in the prefetch buffer if the prefetch assist cache hit signal indicates the predicted address corresponding to the address is not stored in the prefetch assist cache;

means for fetching a next data line corresponding to the predicted address from the main memory and storing the next data line in the prefetch buffer if the prefetch assist cache hit signal indicates the predicted address corresponding to the address is stored in the prefetch buffer;

means for storing the address in the prefetch assist cache such that the address is selected by a last address received from the main processor.

9. The apparatus of claim 7, further comprising:

means for receiving a program counter address, a virtual address, and a load instruction control signal from the main processor;

means for storing the program counter address and the virtual address if the load instruction control signal is received from the main processor, such that a last "n" program counter addresses and virtual addresses are stored;

means for comparing the program counter address to each of the last "n" program counter addresses;

means for generating a predicted virtual address by multiplying a corresponding stored virtual address by 2, and adding the virtual address, if the program counter address equals one of the last "n" program counter addresses;

means for fetching the next data line corresponding to the predicted virtual address from the main memory and storing the next data line in the prefetch buffer, if the program counter address equals one of the last "n" program counter addresses.

10. A cache subsystem, comprising:

control means coupled to receive an instruction prefetch hit signal, a data prefetch hit signal, a main cache hit signal, the control means coupled to receive a processor control signal from a main processor and coupled to receive a memory control signal from a main memory, the control means generating a fill instruction signal, a fill data signal, a fill cache signal, and a next address select signal;

instruction prefetch means coupled to receive an address and the fill instruction signal, the instruction prefetch means storing a plurality of prefetched instruction lines, each of the prefetched instruction lines having a corresponding tag, the instruction prefetch means generating the instruction hit signal if the address corresponds to the tag of one of the prefetched instruction lines, the instruction prefetch means coupled to transmit and receive the prefetched instructions over a data bus;

data prefetch means coupled to receive the address and the fill data signal, the data prefetch means storing a plurality of prefetched data lines, each of the prefetched data lines having a corresponding tag, the data prefetch means generating the data hit signal if the address corresponds to the tag of one of the prefetched data lines, the data prefetch means coupled to transmit and receive the prefetched data lines over the data bus;

main cache means coupled to receive the address and the fill cache signal, the main cache means storing a plurality of unpredicted lines, each of the unpredicted lines having a corresponding tag, the main cache means generating the cache hit signal if the address corresponds to the tag of one of the unpredicted lines, the main cache means coupled to transmit and receive the unpredicted lines over the data bus;

prediction means coupled to receive the address and the next address select signal, the prediction means generating a predicted address, the prediction means selectively transmitting the address and the predicted address to the main memory;

memory interface means coupled to transfer information between the main memory and the main processor, the instruction prefetch means, the data prefetch means, and the main cache means.

11. The cache subsystem of claim 10, wherein the instruction prefetch means comprises fully associative buffer means having a plurality of line entries, the line entries having a data field for storing the prefetched instruction lines and a tag field for selecting the prefetched instruction lines.

12. The cache subsystem of claim 10, wherein the data prefetch means comprises fully associative buffer means having a plurality of line entries, the line entries having a data field for storing the prefetched data lines and a tag field for selecting the prefetched data lines.

13. The cache subsystem of claim 10, wherein the cache means comprises set associative cache means having a plurality of line entries, the line entries having a data field for storing the unpredicted lines and a tag field for selecting the unpredicted lines.

14. The cache subsystem of claim 10, wherein the prediction means comprises:

increment register means coupled to receive the address, the increment register means generating a next sequential address by incrementing the address;

multiplexer means coupled to receive the address, the next sequential address, and the next address select signal, the multiplexer means selectively transmitting the address and the next sequential address to the main memory under control of the next address select signal.

15. The cache subsystem of claim 10, wherein the prediction means comprises:

increment register means coupled to receive the address the increment register means generating a next sequential address by incrementing the address;

prefetch assist means coupled to receive the address and coupled to receive a create PAC entry signal from the control means, the prefetch assist means storing a plurality of predicted addresses, each of the predicted addresses having a corresponding tag, the prefetch assist means generating a PAC hit signal if the address corresponds to the tag of one of the predicted addresses, the prefetch assist means coupled to transmit a next predicted address corresponding to the address if the PAC hit signal is generated;

multiplexer means coupled to receive the address, the next sequential address, the next predicted address, and the next address select signal, the multiplexer means selectively transmitting the address, the next sequential address, and the next predicted address to the main memory under control of the next address select signal.

16. The cache subsystem of claim 15, wherein the prefetch assist means comprises:

set associative cache means having a plurality of line entries, the line entries having a data field for storing the predicted addresses and a tag field for selecting the predicted addresses;

first register means coupled to receive the address and store the address if the processor control signal indicates an instruction read cycle;

second register means coupled to receive the address and store the address if processor control signal indicates a data read cycle.

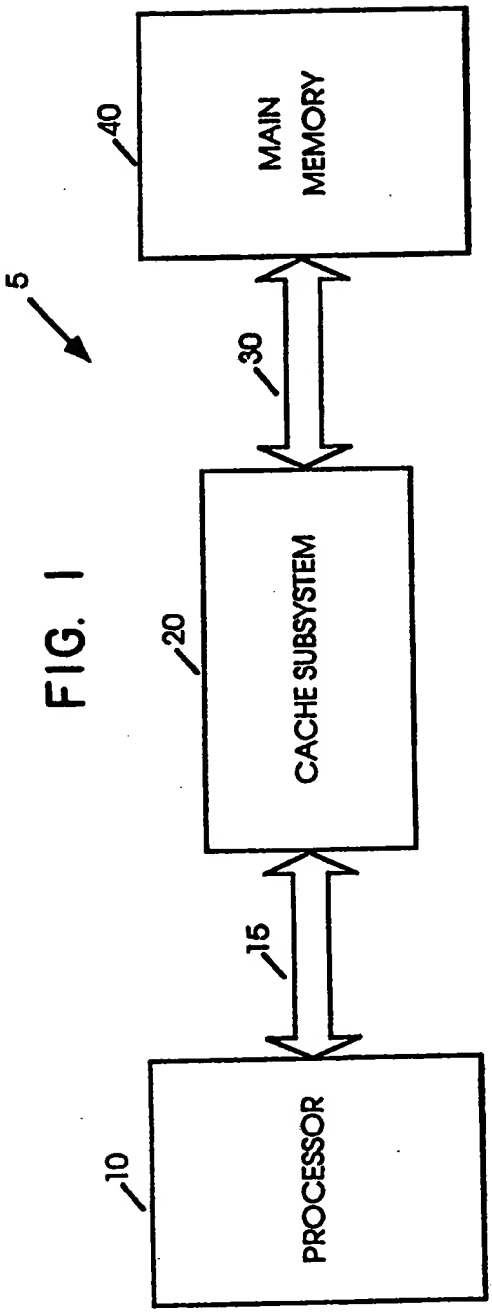

5
40
MAIN MEMORY
30
20
CACHE SUBSYSTEM
15
10
PROCESSOR

FIG. 1

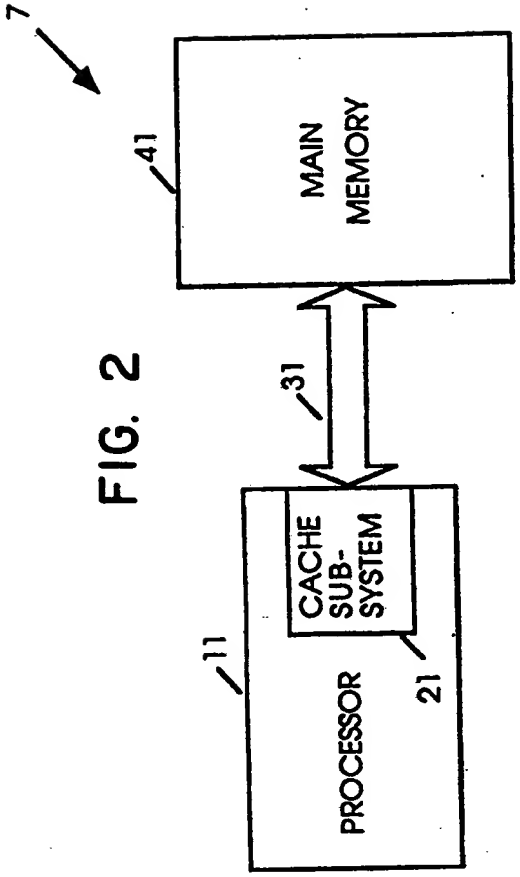

7
41
MAIN MEMORY
31
CACHE SUB-SYSTEM
11
21
PROCESSOR

FIG. 2

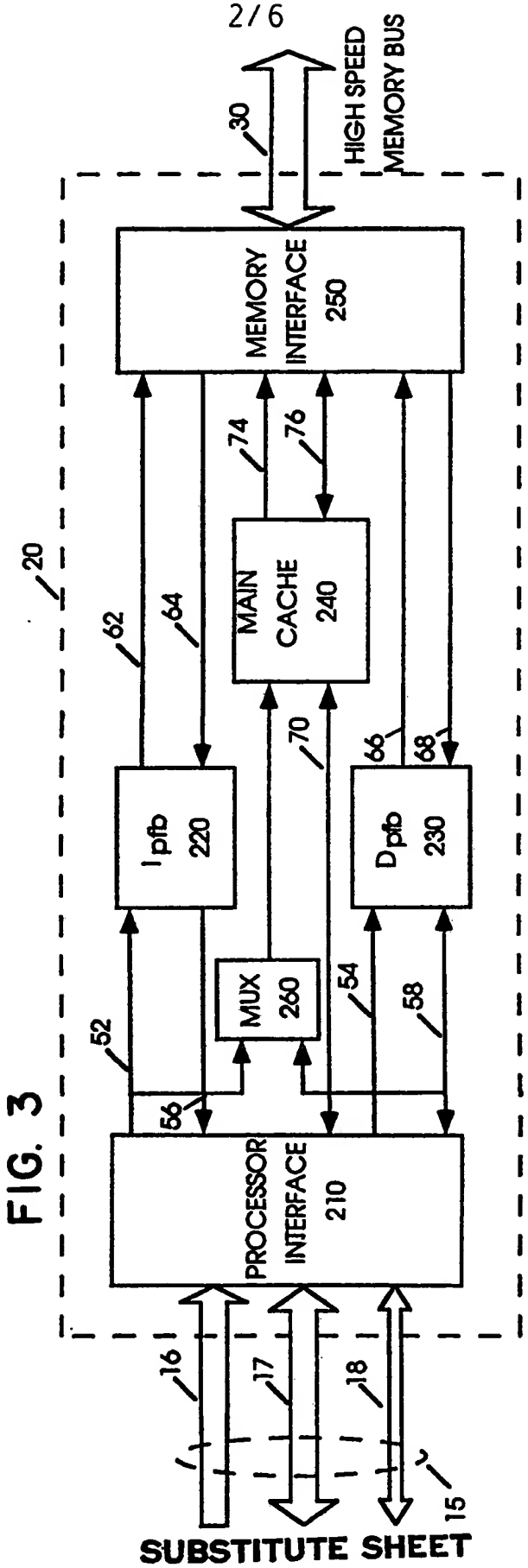
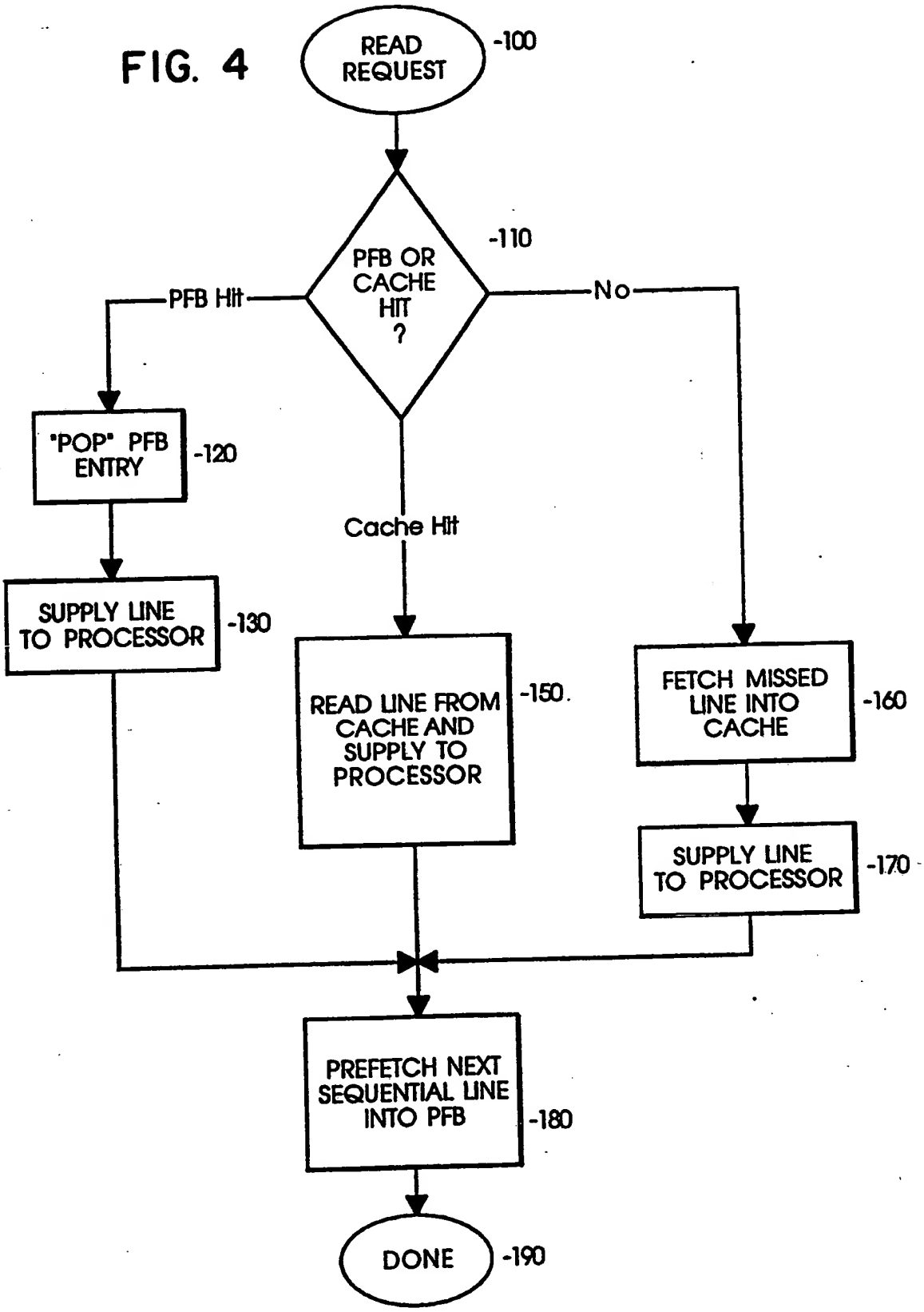

HIGH SPEED MEMORY BUS
30
MEMORY INTERFACE 250
74
76
MAIN CACHE 240
20
62
64
70
66
68
Ipfb 220
Dpfb 230
MUX 260
54
58
52
56
PROCESSOR INTERFACE 210
16
17
18
15

FIG. 3

FIG. 4


READ REQUEST
-100
PFB OR CACHE HIT ?
-110
PFB Hit
No
Cache Hit
"POP" PFB ENTRY
-120
SUPPLY LINE TO PROCESSOR
-130
READ LINE FROM CACHE AND SUPPLY TO PROCESSOR
-150
FETCH MISSED LINE INTO CACHE
-160
SUPPLY LINE TO PROCESSOR
-170
PREFETCH NEXT SEQUENTIAL LINE INTO PFB
-180
DONE
-190

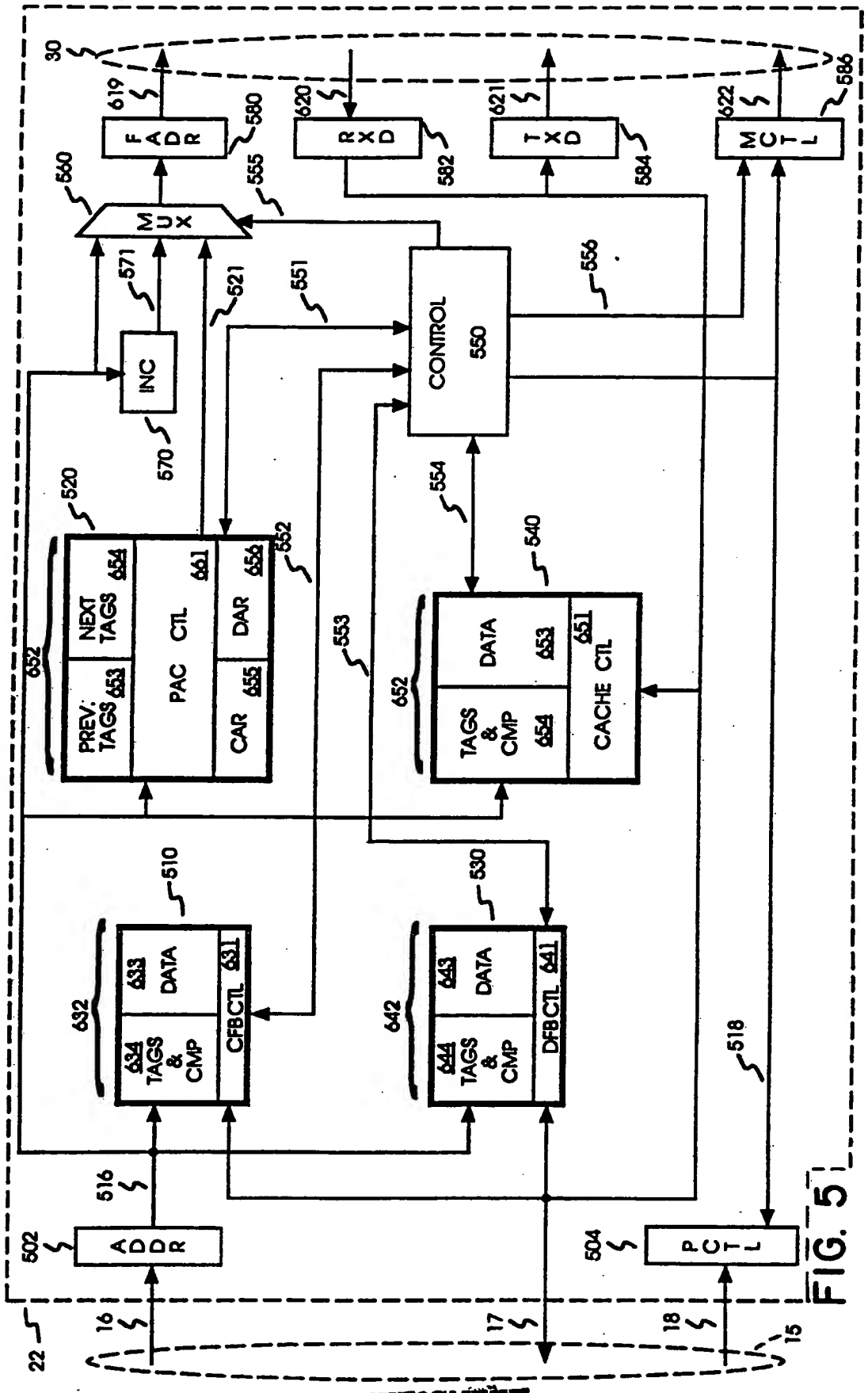
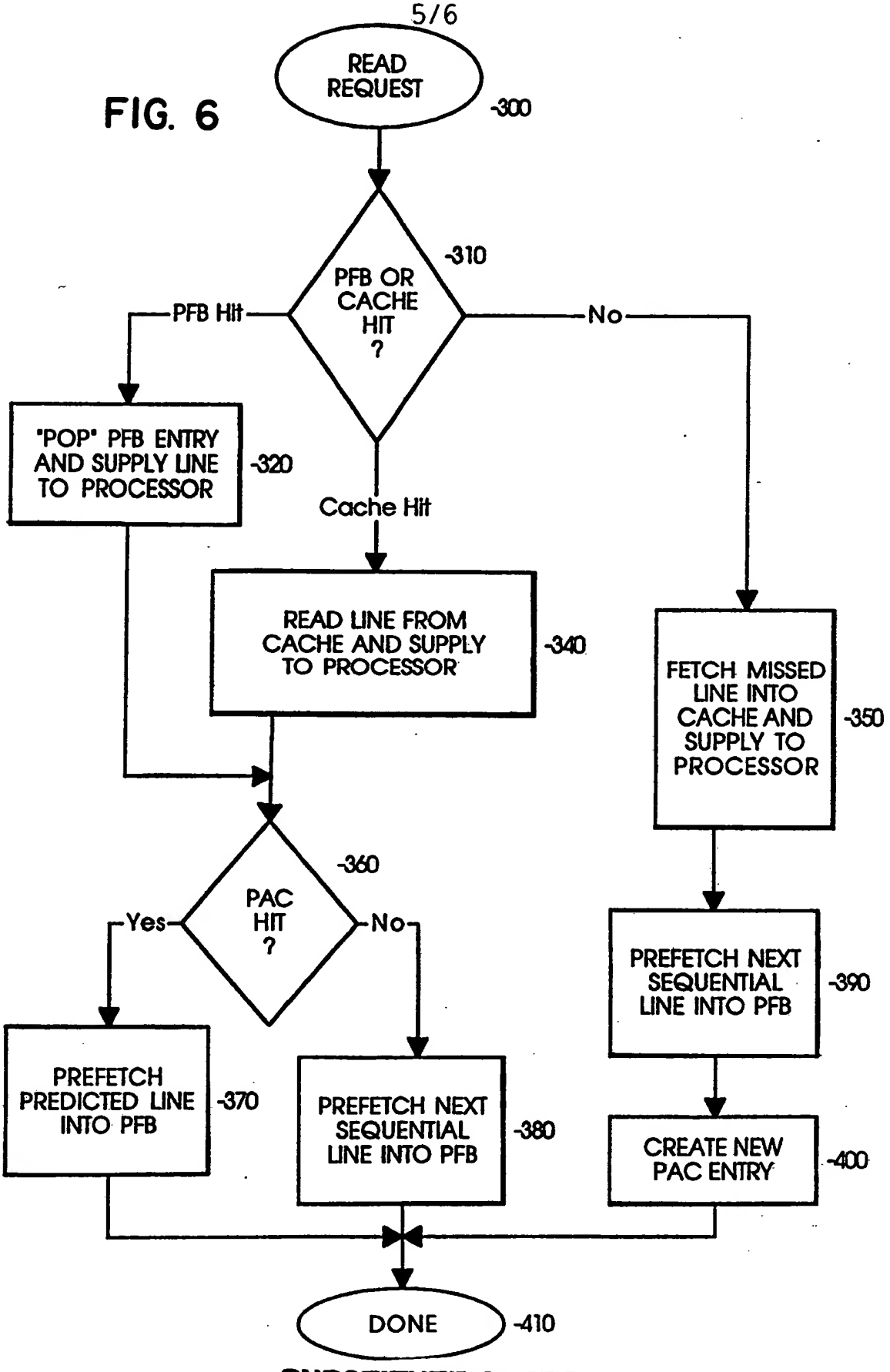

30
619
FADR
580
620
RXD
582
621
TXD
584
622
MCTL
586
560
MUX
555
571
INC
570
521
551
CONTROL
550
556
520
NEXT TAGS
654
PREV. TAGS
653
PAC CTL
651
DAR
656
CAR
655
652
554
540
DATA
653
TAGS & CMP
654
CACHE CTL
651
652
553
510
DATA
633
TAGS & CMP
634
CFBCTL
631
632
530
DATA
643
TAGS & CMP
644
DFBCTL
641
642
518
516
502
ADDR
504
PCTL
22
16
17
18
15

FIG. 5

FIG. 6


READ REQUEST
-300
PFB OR CACHE HIT ?
-310
PFB Hit
No
Cache Hit
"POP" PFB ENTRY AND SUPPLY LINE TO PROCESSOR
-320
READ LINE FROM CACHE AND SUPPLY TO PROCESSOR
-340
FETCH MISSED LINE INTO CACHE AND SUPPLY TO PROCESSOR
-350
PAC HIT ?
-360
Yes
No
PREFETCH PREDICTED LINE INTO PFB
-370
PREFETCH NEXT SEQUENTIAL LINE INTO PFB
-380
PREFETCH NEXT SEQUENTIAL LINE INTO PFB
-390
CREATE NEW PAC ENTRY
-400
DONE
-410

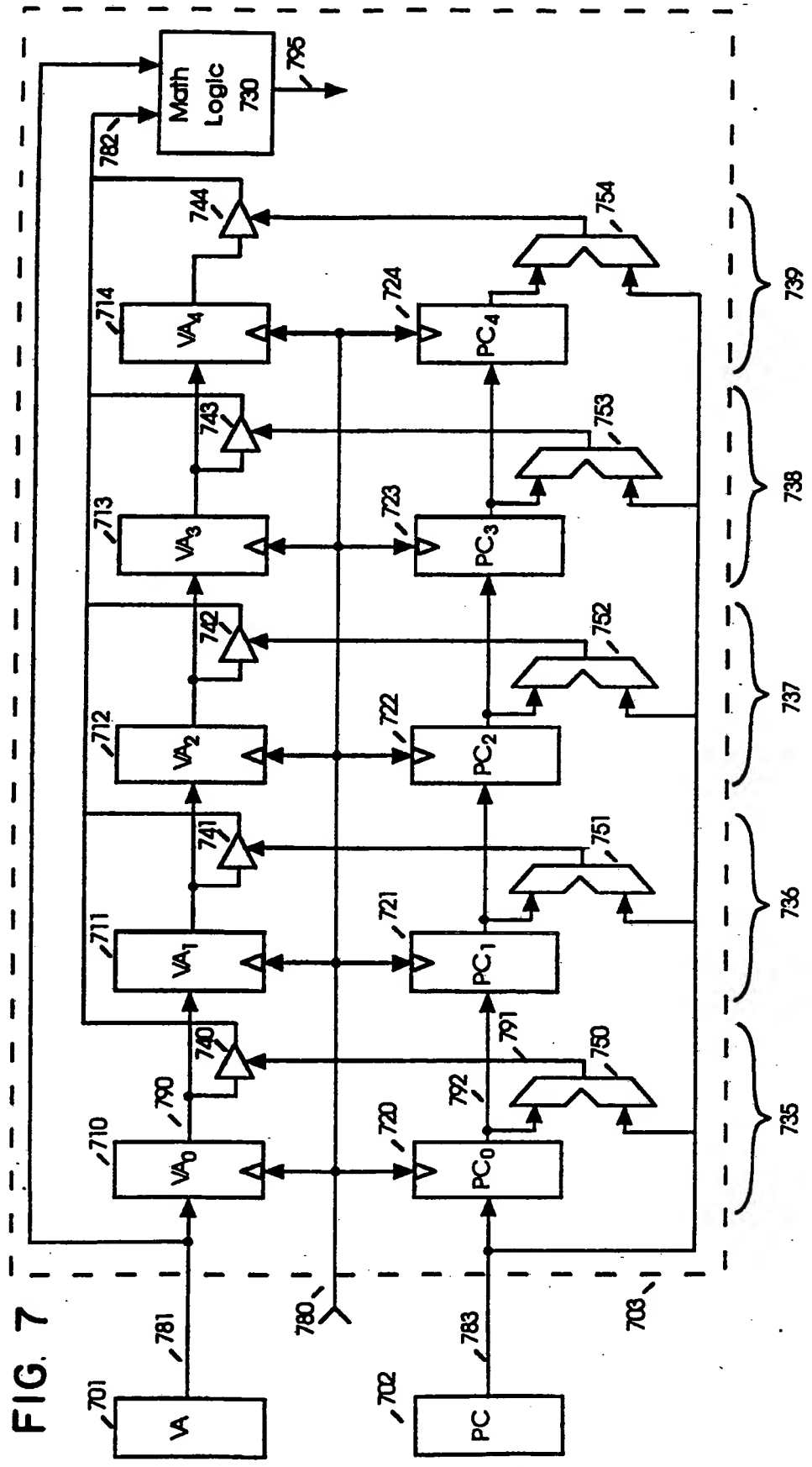

Math Logic 730
795
782
744
754
714
VA4
724
PC4
739
743
753
713
VA3
723
PC3
738
742
752
712
VA2
722
PC2
737
741
751
711
VA1
721
PC1
736
740
791
750
790
792
710
VA0
720
PC0
735
781
780
783
703
701
VA
702
PC

FIG. 7

INTERNATIONAL SEARCH REPORT

International Application No PCT/US 93/01814

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all)[6]

According to International Patent Classification (IPC) or to both National Classification and IPC

Int.Cl. 5 G06F12/08

II. FIELDS SEARCHED

Minimum Documentation Searched[7]

Classification System	Classification Symbols
Int.Cl. 5	G06F

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched[8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT[9]

Category °	Citation of Document,[11] with indication, where appropriate, of the relevant passages[12]	Relevant to Claim No.[13]
X	ELEKTRONISCHE RECHENANLAGEN - MIT COMPUTER PRAXIS vol. 15, no. 2, 1973, MUNCHEN DE pages 60 - 65 SWOBODA 'Möglichkeiten der Beschleunigung einer Zentraleinheit durch strukturelle Massnahmen' see page 62, paragraph 2.2	1-3,7
Y	EP,A,0 449 540 (DIGITAL EQUIPMENT CORP.) 2 October 1991 see abstract see page 9, line 37 - page 13, line 41; figures 13,16,19	1-5,7,8, 10-15

-/--

° Special categories of cited documents :[10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
25 MAY 1993	11. 06. 93
International Searching Authority	**Signature of Authorized Officer**
EUROPEAN PATENT OFFICE	NIELSEN O.P.

Form PCT/ISA/210 (second sheet) (January 1985)

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)

Category°	Citation of Document, with indication, where appropriate, of the relevant passages	Relevant to Claim No.
Y	EP,A,0 457 403 (N.V. PHILIPS GLOEILAMPENFABRIEKEN) 21 November 1991 see column 3, line 3 - column 4, line 4 see column 5, line 10 - column 6, line 10	1-5,7,8, 10-15
Y	IBM TECHNICAL DISCLOSURE BULLETIN vol. 34, no. 2, July 1991, NEW YORK US 'Algorithm for Non-Sequential Cache Prefetching' see the whole document	4,5,8,15 16
A	US,A,5 007 011 (MURAYAMA) 9 April 1991 see abstract see column 3, line 36 - column 5, line 68; figures 2A-2C	6,9

ANNEX TO THE INTERNATIONAL SEARCH REPORT ON INTERNATIONAL PATENT APPLICATION NO.

US 9301814
SA 71624

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information. 25/05/93

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP-A-0449540	02-10-91	JP-A- 4270431	25-09-92
EP-A-0457403	21-11-91	JP-A- 4232549	20-08-92
US-A-5007011	09-04-91	JP-A- 62035949	16-02-87

EPO FORM P0479

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82